How to / Maniaplanet / Manialink / Build a Manialink

Author : Marcel @ Funtrackers

LAST PAGE PAGE 1 2 3 4 5 6 7 8 9 NEXT PAGE

5. Tips and Tricks

Selected Topics which are Worth Knowing

The last chapter of this Tutorial will explain some selected topics, which either do not match into the other chapters, or simply are too much to write them there. The following parts are not related to each other, so you may read those parts which you are interested in, and skip the others.

5.1 Advanced Example of Entry and FileEntry

PHP knowledge required.

With Entrys and FileEntrys the user has the possibility to send entered values and files to the ManiaLink. This section will now explain with a more advanced example, how you have to use these elements and how you finally get your hands on the transmitted data.

Advanced example for Entry

The use of an Entry is very easy, as the values can directly be appended to the URL as additional parameters.

XML code:

```
<entry sizen="10 5" style="TextValueSmall" name="inputValue" default="Hello World!" />
<label posn="0 -20" style="CardButtonMedium" text="Link with value" manialink="ManiaLink?value=inputValue" />
```

Handling script:

```
<?php
// Read the passed value
$value = $_GET['value'];

// Escape the value to make our ManiaLink safe against manipulation
$value = htmlspecialchars($value);

// Output the ManiaLink
header('Content-Type: text/xml');
echo <<<EOT
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1">
<timeout>0</timeout>
<label text="{$value}" />
</manialink>
EOT;
?>
```

In the XML file, we have an <entry> with the name inputValue. Whereever this name appears in a URL of a link, it is replaced with the current value of the Entry. So in our case, the inputValue of the link in the Label is replaced by whatever the user entered into the Entry. Or in other words: The content of the Entry is passed as parameter with the name value to our script, as specified with ?value=inputValue.

In the script we now simply have to read the passed parameter using the $_GET array, and afterwards generate a full ManiaLink as output, where the passed value is printed as label.

Advanced example for FileEntry

The use of a FileEntry is a little bit more complex, as we cannot pass the file as URL parameter, but have to POST it.

XML code:

```
<fileentry sizen="50 5" style="TextValueSmall" name="inputMap" folder="Maps" default="Select Map..."/>
<label posn="0 -20" style="CardButtonMedium" manialink="POST(ManiaLink?map=inputMap,inputMap)" text="Send Map"/>
```

Handling script:

```
<?php
// Read the passed file
$name = $_GET['map'];
$file = file_get_contents('php://input');

// Save the passed file on the server
// WARNING: Always check the file uploaded by the user to be what you expected.
// This is not done in this example, what is a huge security risk!

file_put_contents("maps/{$name}", $file);

// Get the filesize for outputting it
$size = filesize("maps/{$name}");

// Escape the filename
$name = htmlspecialchars(\$name);

// Output the ManiaLink
header('Content-Type: text/xml');
echo <<<EOT
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1">
<timeout>0</timeout>
<label text="Received Map: {$name} (Size: {$size})" />
</manialink>
EOT; ?>
```

The XML code is very similar to that one for the Entry example, the only important different is the value of the manialink attribute of the label. Instead of having the ManiaLink to be called, we now see a POST(ManiaLink?map=inputMap,inputMap). With this we now send the file of the FileEntry inputMap (behind the comma) to the ManiaLink ManiaLink?map=inputMap (infront the comma). But why dies the FileEntry occurs twice in this link? The reason is, that we will loose the filename if we only send the file itself. But as we need the filename to save the file on the server, we have to pass it separatly to the script. (This also means, if you do not need the filename itself, you may omit the parameter in the URL, making it to a POST(ManiaLink,inputMap)).

The script itself now reads the filename and the file contents, saves the file on the server and prints the name and the size of the received file as new ManiaLink. The code is not very difficult once we managed to send out file using a POST request.

Hint: Another time I want to state, that you always should check what file is uploaded to the server! For our example, a simply check would be to validate the filename to end with .Map.Gbx, a better but more difficult to implement security check would be to read and validate the XML header of the Map.

5.3 ManiaConnect authentification

PHP knowledge required.

ManiaConnect is a system, with which you are able to retrieve detailed information of the player currently visiting the ManiaLink – assuming that he granted access to his or her data. These information are secure and cannot be manipulated at all, so if you have the login m4rcel, you can be sure that I am visiting your ManiaLink (or somebody hacked my account).

ManiaConnect is part of the ManiaPlanet Web Services (MPWS). The following preconditions must be fulfilled by your webspace to be able to use these services and ManiaConnect in particular:

PHP 5.3 or newer
CURL extension
JSON extension

If your webspace supports these requirements, go ahead and download the latest version of the **MPWS SDK** for PHP.

Creating an API user for the MPWS

Before we can request any data, we have to create a new API user with which we will be able to use the MPWS. Creating a new API user is very simple:

1) Visit `developers.maniaplanet.com` and login with your ManiaPlanet account.
2) Click on Web Services at the top of the page, and use the button Create a new API user.
3) Do what the page prompts you to do, until you have created the API user, landing on the Manage your API user page.
4) Click on the button Create a ManiaConnect application: Again, the dialog is self-explanatory in what values are to be entered. Important is the Redirection URI: This is the URL, which should be redirected to after the user permitted access to his or her data.
5) Done.

Request Data with ManiaConnect

Now that we have created our API user and a ManiaConnect Application for it, we are ready to write a ManiaLink which uses the MPWS to display the user's information. The example below demonstrates this by printing the Nickname of the current user.

ManiaLink using ManiaConnect to show the Nickname:

```
<?php

// Change these values to match your MPWS user

define('MPWS_USER', 'mpws_user');
define('MPWS_PASS', 'mpws_pass');
define('MPWS_SCOPE', 'basic');

// Include the MPWS files
require('autoload.php');
// Try to retrieve the data of the player
$mpwsPlayer = new \ManiaPlanet\WebServices\ManiaConnect\Player(MPWS_USER, MPWS_PASS);
$player = $mpwsPlayer->getPlayer();

if (!$player) {
// If $player equals false, we need the player to grant access to his or her data,
// so we need to print the LoginURL
$loginURL = htmlspecialchars($mpwsPlayer->getLoginURL(MPWS_SCOPE));
$label = '<label style="CardButtonMedium" text="Authentificate" manialink="'.$loginURL.'" />'; }
else {
// If $player is not false, the player allowed accessing his or her data,
// so we are able to e.g. print the Nickname
$nickname = htmlspecialchars($player->nickname);
$label = '<label sizen="100 5" halign="center" text="Your Nickname: '.$nickname.'" />'; }

// Output the ManiaLink
header('Content-Type: text/xml; charset=utf-8');
echo <<<EOT <?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1">
<timeout>0</timeout>
{$label}
</manialink>
EOT; ?>
```

Most important is that the Player class is instantiated before the first output, because it will initialize a session (using the PHP's internal session mechanisms). You then simply try to access the player's information using the getPlayer() method: If you are allowed to do so, you will get the data of the player, if not, false is returned instead. According to the returned value, we therefore are able to decide if we already can print the Nickname, or if we have to get the permission first.

If we don't have the permission to get the player's data, we need to call getLoginURL() to get the URL to the ManiaLink on which the player can grant access to his or her data. We do not take care of this, simply print this URL e.g. as button, and "wait" that the user is redirected back to our ManiaLink: If he allowed accessing the data, we will get them and thus printing the Nickname, otherwise the button is displayed another time.

The Scopes of ManiaConnect

One thing, which has not been explained so far, is left: The scopes of ManiaConnect. The scope simply says which data you want to get access to, and therefore which data the player must permit to be accessed.

List of available scopes

- basic – Method: getPlayer() – Basic information like login, nickname and zone
- buddies – Method: getBuddies() – List of the buddies
- dedicated – Method: getDedicated() – List of the dedicated server
- email – Method: getEmail() – Email address
- manialinks – Method: getManiaLinks() – List of the registered ManiaLink codes
- online_status – Method: getOnlineStatus() – Online status

If using several scopes, simply separate them with spaces.

Hint: Do only request those scopes you really need access to. For example, a Shoutbox or a Guestbook should never ask to get the buddies of a player.

Additional information

If you want to revoke any permission of a certain ManiaLink, you can visit home.maniaplanet.com, login with your ManiaPlanet account, and manage all applications which currently have access to your data.

LAST PAGE PAGE 1 2 3 4 5 6 7 8 9 NEXT PAGE

This site is friendly to stationary (PC/MAC) and mobile (Android/iOS) browsers
Trackmania, Shootmania, Questmania, games & media © Nadeo & UbiSoft used under fair use policy
8,373,041 page views since January 2008. All tutorials, services, coding © Mania Creative 2006-2013